

# ZJU Summer 2024 Training

## Group B Contest 1

memset0 zzw4257 a999999 water\_tomato Hydroxythio

2024-07-07

Zhejiang University

## Black and White 1

给定一个  $n \times m$  的棋盘，其中部分位置有棋子。你可以依次进行以下操作：

- 选中一个棋子  $(x, y)$ ，满足其至少一个边相邻的棋子。
- 选中一个棋子  $(x', y')$ ，其与前面选定的棋子  $(x, y)$  边相邻。
- 将棋子  $(x, y)$  从棋盘中拿走。如果棋子  $(x, y)$  是白色则将棋子  $(x', y')$  反色。

判断最后能否只剩下一枚棋子且该棋子的颜色为白色。


## 题解

将白色棋子的权重视为 1，黑色棋子的权重视为 0。则每次操作不影响所有棋子权重的异或和。对于初始局面，我们可以直接检查异或和是否为 1。

## 题解

将白色棋子的权重视为 1，黑色棋子的权重视为 0。则每次操作不影响所有棋子权重的异或和。对于初始局面，我们可以直接检查异或和是否为 1。

真的这么简单吗？



## 题解

将白色棋子的权重视为 1，黑色棋子的权重视为 0。则每次操作不影响所有棋子权重的异或和。对于初始局面，我们可以直接检查异或和是否为 1。

我们还需要检查，所有棋子是否在同一个联通块中。



真的这么简单吗？

## Black and White 2

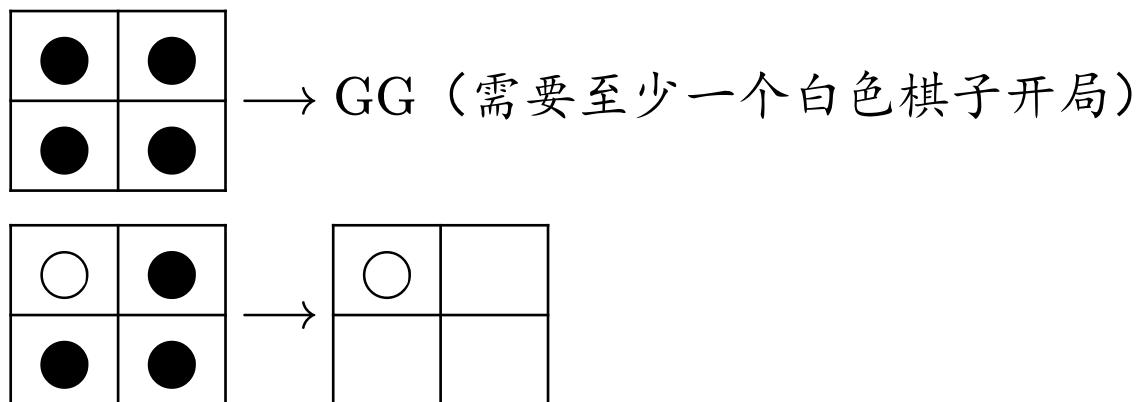
给定一个  $n \times m$  的棋盘，其中部分位置有棋子。你可以依次进行以下操作：

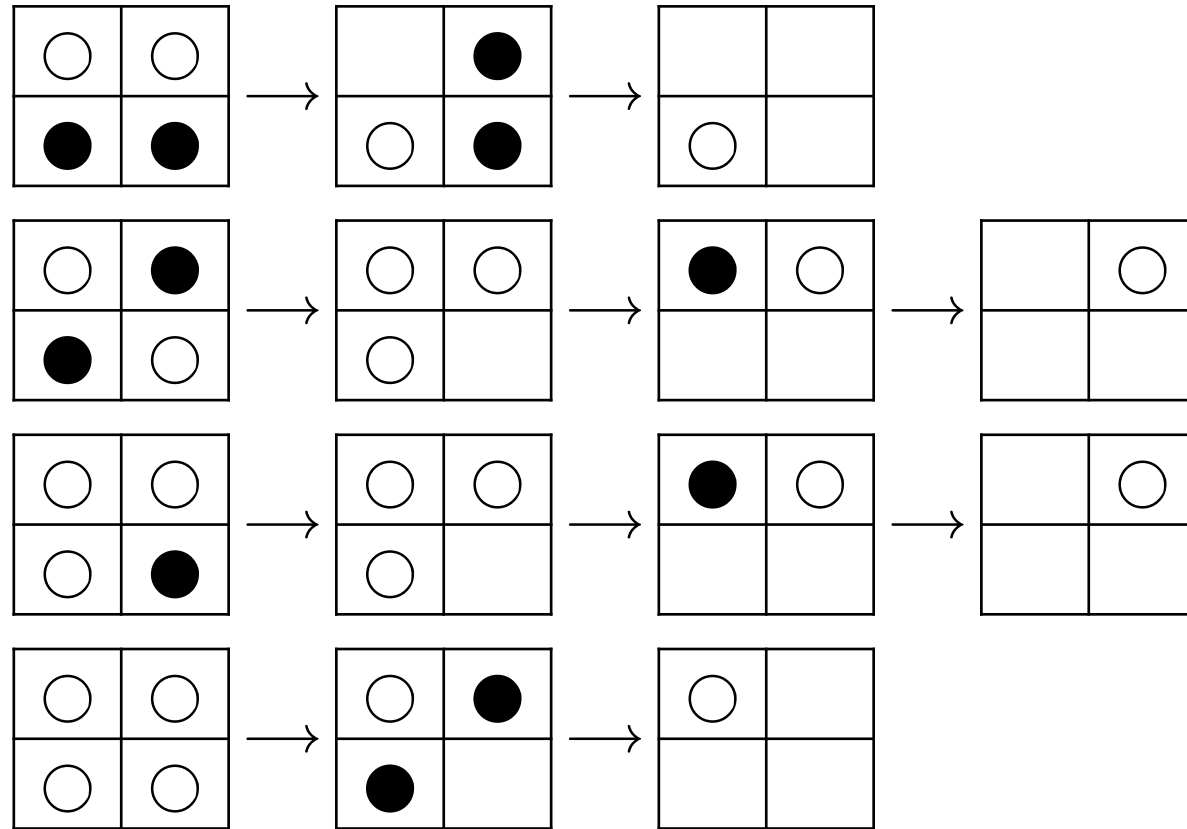
- 选中一个棋子  $(x, y)$ ，满足其至少一个边相邻的棋子。
- 将棋子  $(x, y)$  从棋盘中拿走。如果棋子  $(x, y)$  是白色则将棋子所有边相邻的棋子  $(x', y')$  反色。

判断最后能否只剩下一枚棋子且该棋子的颜色为白色。

## 手玩魅力时刻

可以证明，对于  $2 \times 2$  棋盘的情况，只要存在白色棋子，就一定会成功。







## 归纳到简单情况

对于  $n \times m$  的情况 ( $n > 2$  或  $m > 2$ ), 我们希望将其转换为带有一个白色棋子的  $2 \times 2$  情况, 这让我们想到把外侧的棋子先拿到, 并将一个白色的棋子“保护”起来。

## 归纳到简单情况

对于  $n \times m$  的情况 ( $n > 2$  或  $m > 2$ ), 我们希望将其转换为带有一个白色棋子的  $2 \times 2$  情况, 这让我们想到把外侧的棋子先拿到, 并将一个白色的棋子“保护”起来。

找到最上方的白色棋子 (如果有多个则选择最左侧的), 如果这个棋子位于最右边一列或最下边一行则可以考虑旋转棋盘或微调一下策略。

## 归纳到简单情况

对于  $n \times m$  的情况 ( $n > 2$  或  $m > 2$ ), 我们希望将其转换为带有一个白色棋子的  $2 \times 2$  情况, 这让我们想到把外侧的棋子先拿到, 并将一个白色的棋子“保护”起来。

找到最上方的白色棋子 (如果有多个则选择最左侧的), 如果这个棋子位于最右边一列或最下边一行则可以考虑旋转棋盘或微调一下策略。

将这个棋子上方的所有棋子都拿掉 (一定全是黑色的), 之后将这一棋子同一行中左侧的棋子都拿掉 (也一定全是黑色的)。之后按照任意顺序拿走其除了以这一棋子为左上角的  $2 \times 2$  区域外的所有棋子。则这一棋子的颜色不会被反转。

## 归纳到简单情况

对于  $n \times m$  的情况 ( $n > 2$  或  $m > 2$ ), 我们希望将其转换为带有一个白色棋子的  $2 \times 2$  情况, 这让我们想到把外侧的棋子先拿到, 并将一个白色的棋子“保护”起来。

找到最上方的白色棋子 (如果有多个则选择最左侧的), 如果这个棋子位于最右边一列或最下边一行则可以考虑旋转棋盘或微调一下策略。

将这个棋子上方的所有棋子都拿掉 (一定全是黑色的), 之后将这一棋子同一行中左侧的棋子都拿掉 (也一定全是黑色的)。之后按照任意顺序拿走其除了以这一棋子为左上角的  $2 \times 2$  区域外的所有棋子。则这一棋子的颜色不会被反转。

这样, 我们就将题目转化到了前面  $2 \times 2$  的情况, 复用上面的构造即可。

## Mutually Orthogonal Latin Squares

构造一个  $n \times n$  的值域为  $[1, n] \cap \mathbb{Z}$  的矩形, 使得每行每列和主对角线元素互不相同。

## 题解

观察发现奇数情况很简单，取  $a_{i,j} = ((i + j - 2) \bmod n) + 1$  即可；

大于 2 的偶数注意上述构造中次对角线互不相同（循环意义上），将其映射到扩增的第  $n$  行和第  $n$  列，将空出来的位置填上  $n$ 。

## 题解

观察发现奇数情况很简单，取  $a_{i,j} = ((i + j - 2) \bmod n) + 1$  即可；

大于 2 的偶数注意上述构造中次对角线互不相同（循环意义上），将其映射到扩增的第  $n$  行和第  $n$  列，将空出来的位置填上  $n$ 。

容易证明只有  $n = 2$  无解。

## Crush

$N$  个人，每一个人有一个暗恋对象，站在其右边第一个位置就能满足，排列的价值定义为满足的人的数量。求排列的最大价值和能达到最大价值的排列个数。

$3 \leq N \leq 10^6$ 。



## 题解

首先把人抽象成点，暗恋关系抽象成有向边。每个点有一条出边，那么就构成了一个内向树森林。

我这里把边反一下，从每个点有一条指向其父亲的边，变成父亲指向所有的儿子。这样理解的话，就是外向树森林了。

不同的树之间相互独立，所以可以对每棵树计算两问的答案，问一相加、问二相乘即可。

接下来的方法，从观察一个人能不能让别人满意的角度来分析。

## 题解

现在只考虑一棵外向树。

先不考虑环，我们看看一棵正常的树怎么做。对于一个点  $x$ ，它让别人满意的方式，就是站在其某一个儿子的左边第一个位置。如果树有  $n$  个节点， $k$  个叶子，那么其他的  $n - k$  个内部节点一定恰有 1 的贡献，所以问一的答案为  $n - k$ 。同时也可以发现，无论  $x$  站在哪个儿子后面，不影响最终的贡献大小。所以问二的答案为  $\prod_x c_x$ ， $x$  表示非叶子节点， $c_x$  表示  $x$  的儿子个数。

## 题解

再来考虑环。

先来点特殊情况，只有一个长度为  $l$  的环，不存在其他节点。那么从任意一个点开始，沿着树边排好，得到的。第一问的答案就是  $l - 1$ ，第二问的答案为  $l$ 。

然后给一个节点  $x$  上加几个子树。如果环上排列不以  $x$  结尾，那么结尾总会有一个点不能让别人满意（因为接下来就是  $x$  的子树的部分），但以  $x$  结尾就可以。所以这里必须要以  $x$  结尾，答案就为  $l + (x \text{ 为根的正常树的答案} - 1)$ 。问二的答案就是  $1 \times (x \text{ 为根的正常树的答案})$ 。注意算正常树的时候不要带上环上的边。

## 题解

如果有多个节点有子树呢？那么它们其中任意一个是环上排列结尾即可。注意，对于所有的正常子树，要去掉根的贡献，因为它的根不在任何儿子的前面了，而在的那个贡献算到前面来了。设有子树的点集为  $S$ ，则问一答案为  $l + \sum_{x \in S} (x \text{ 为根的正常树的答案} - 1)$ 。

问二的情况要特殊一些。

通过上一个情况的分析，我们可以发现要满意的人最多，环上排列的每一个点，它的后一个点要么是环上的下一个点，要么是环外的一个儿子。只有一种情况是不合法的：所有点都选择下一个点还在环上。那么问二的答案就是  $(\prod_{x \in S} (c_x + 1) - 1) \times \prod_y$ （以  $y$  为根的答案），其中  $y$  的父亲在  $S$  中（也就是在环上）。

## 题解

如果有多个节点有子树呢？那么它们其中任意一个是环上排列结尾即可。注意，对于所有的正常子树，要去掉根的贡献，因为它的根不在任何儿子的前面了，而在那个贡献算到前面来了。设有子树的点集为  $S$ ，则问一答案为  $l + \sum_{x \in S} (x \text{ 为根的正常树的答案} - 1)$ 。

问二的情况要特殊一些。

通过上一个情况的分析，我们可以发现要满意的人最多，环上排列的每一个点，它的后一个点要么是环上的下一个点，要么是环外的一个儿子。只有一种情况是不合法的：所有点都选择下一个点还在环上。那么问二的答案就是  $(\prod_{x \in S} (c_x + 1) - 1) \times \prod_y$ （以  $y$  为根的答案），其中  $y$  的父亲在  $S$  中（也就是在环上）。

最后考虑一下，产生贡献的边将点都连了起来，没有连边的链之间的位置可以随意交换。有多少条链呢？ $n - ans_1$ 。所以最后问二的答案要乘以  $(n - ans_1)!$ 。

## Beryl and Sapphire

用若干个立方体积木堆一个形体，给定左视图的所有高度  $\{h_i\}$  和最小需要花费的积木数量  $k$ ，试构造一个主视图。

保证  $h_i$  互不相同且  $\sum h_i \leq k \leq \sum h_i + (n - 2) \times \max\{h_i\}$ 。

## 题解

不妨先考虑如果已知主视图和左视图，如何求最小的积木数。

记主视图和左视图的高度分别为  $\{a_i\}$  和  $\{b_i\}$ ，首先我们有  $\max\{a_i\} = \max\{b_i\}$ ，否则一定不存在合法的解。由于主视图和左视图的高度顺序和最小积木数无关，我们不妨令  $a_n = b_n = \max\{a_i\}$ 。然后，一个简单的想法是我们令第  $n$  行的格子上放的积木高度分别为  $a_1, a_2, \dots, a_n$ ，第  $n$  列的格子上放的积木高度为  $b_1, b_2, \dots, b_n$ 。当然，这不是最优解，对于其中的  $a_i = b_j$ ，我们可以改成在第  $i$  列第  $j$  行的格子上堆一个高度为  $a_i$  的积木。（以上即为 checker 写法）

## 题解

因此我们有一个构造思路，不妨令  $b_n$  为最大值，其余随意，令  $a_i = \max\{b_i\}$ ，我们就找到了最大的最小需要积木数，然后我们从 1 枚举到  $n - 1$ ，一直令  $a_i = b_i$ ，直至与要求的  $k$  的差距不足  $\max\{b_i\}$ ，记这个差距为  $w$ ，我们只要令  $a_i = \max\{b_i\} - w$  即可。这个时候还要校验一下  $a_i$  在  $b_i \sim b_{n-1}$  中是否存在，若存在，则在之前的  $a_1, a_2, \dots, a_{i-1}$  中随意找一个，也令其的值为  $\max\{b_i\} - w$  即可。

你会发现  $h_i$  互不相同和  $k$  的上界共同保证了这个构造方法是一定有效的。



## Purely Bullet Hell

有一个游戏，包含  $n$  关， $m$  种怪物，第  $i$  种怪物会出现在第  $p_i$  到第  $q_i$  关 ( $p_i \leq q_i$ )，仅在第一次见到第  $i$  种怪物时会损失  $a_i$  点生命值，每通过一关（这意味着见到这一关会出现的所有种类的怪物）则会获得  $b_i$  点生命值。考虑集合  $S_{l,r} = \{x | x \in \mathbb{Z}, l \leq x \leq r\}$ ，定义  $f(l, r)$  为：从  $S_{l,r}$  中选出一个非空子集  $T$ ，只游玩  $T$  中的关卡，所有方案中生命值变化量（所有获得的生命值减去所有损失的生命值）的最大值。 $q$  次询问，每次给出  $l, r$ ，求  $f(l, r)$  的值。

$1 \leq n, m, q \leq 5 \times 10^5, 1 \leq a_i, b_i \leq 10^9$ 。

## 题解

令：

$$g(l, r) = \min_{T \subseteq S_{lr-1}} \left\{ \sum_{[p_i, q_i] \cap (T \cup \{r\}) \neq \emptyset} a_i - \sum_{i \in T \cup \{r\}} b_i \right\}$$

则  $f(l, r) = -\min_{l \leq i \leq r} \{g(l, i)\}$ 。

考虑如何计算  $g(l, r)$ 。我们分两种情况讨论：固定  $l$ ，由  $g(*, r)$  推  $g(*, r+1)$ ；固定  $r$ ，由  $g(l, *)$  推  $g(l+1, *)$ 。

固定  $l$

记  $cnt(i, j) = \sum_{i < p_k \leq j, q_k \geq j} a_k$ , 显然有转移:

$$g(l, r) = \min \left\{ cnt(0, r), \min_{l \leq i < r} \{ g(l, i) + cnt(i, r) \} \right\} - b_r$$

其中,  $cnt(0, r)$  是易于维护的, 而  $g(l, i) + cnt(i, r)$  可以使用线段树维护, 因此可以用  $O(n \log n)$  的时间复杂度算出  $f(1, *)$ 。

## 固定 $r$

这部分主要考虑当  $l$  变为  $l+1$  时,  $g(l+1, r)$  与  $g(l, r)$  的关系。

记  $i$  的最优转移点为  $bp(i)$  (若  $i$  的转移取  $cnt(0, i)$  一项, 则认为  $bp(i) = 0$ ),  $bp^k(i) = \begin{cases} bp(bp^{k-1}(i)) & k > 0 \\ i & k = 0 \end{cases}$ ,  $g(l, i)$  所对应的选择方案的集合为  $BP(i)$ 。注意到转移方程的实际意义是枚举  $BP(r)$  中的次大元素, 因此我们有:  $BP(i) = (\cup_{k \geq 0} \{bp^k(i)\}) \setminus \{0\}$ 。

考虑前一情况转移中的  $\min_{l \leq i < r} \{g(l, i) + cnt(i, r)\}$  的部分, 令  $a < b < c < d$ , 考虑  $g(l, c)$  和  $g(l, d)$  的最优转移点: 有两种情况,  $g(l, c)$  的最优转移点是  $g(l, a)$  (即  $bp(c) = a$ ),  $g(l, d)$  的最优转移点是  $g(l, b)$  (即  $bp(d) = b$ ); 或者相反。分别计算这两种情况中  $g(l, c) + g(l, d)$  的值:

固定  $r$

记  $c(i, j)$  表示左端点在上图编号为  $i$  的区间中，右端点在上图编号为  $j$  的区间中的所有  $a_i$  的和，那么：

转移区间交叉的情况（即  $g(l, a) \rightarrow g(l, c), g(l, b) \rightarrow g(l, d)$ ）

$$\begin{aligned} g(l, c) + g(l, d) &= (g(l, a) + c(2, 4) + c(2, 5) + c(3, 4) + c(3, 5) - b_c) \\ &\quad + (g(l, b) + c(3, 5) + c(4, 5) - b_d) \end{aligned}$$

转移区间包含的情况（即  $g(l, a) \rightarrow g(l, d), g(l, b) \rightarrow g(l, c)$ ）

$$\begin{aligned} g(l, c) + g(l, d) &= (g(l, b) + c(3, 4) + c(3, 5) - b_c) \\ &\quad + (g(l, a) + c(2, 5) + c(3, 5) + c(4, 5) - b_d) \end{aligned}$$

## 固定 $r$

观察上述两种情况,可以发现前一种情况相比后一种情况,多一项  $c(2,4)$ 。若令  $w(i,j) = g(l,i) + cnt(i,j) - b_j$ , 那么:

$$w(a,c) + w(b,d) \geq w(a,d) + w(b,c)$$

恒成立。(注:反向的四边形不等式,熟悉 dp 优化的选手想到这一步是很自然的)

固定  $c,d$ ,  $a$  和  $b$  之间的关系是我们关心的,接下来对此展开讨论。

假设  $a < b < c = r$ , 那么根据最优化条件,  $w(a,c) < w(b,c)$  且  $w(b,d) < w(a,d)$ , 即  $w(a,c) + w(b,d) < w(a,d) + w(b,c)$ , 而这是与上述不等式是矛盾的,因此假设不成立。于是我们得出结论:对于任意的  $c,d$ , 若  $c < d$ , 则要么  $bp(d) \leq bp(c)$ , 要么  $c \leq bp(d) < d$ 。

## 固定 $r$

上述结论有一个重要推论： $bp(i+1) \subseteq BP(i)$ ,  $BP(i+1) \setminus \{i+1\} \subseteq BP(i)$ , 也即存在整数  $n$ , 满足  $bp(i+1) = bp^n(i)$ ,  $BP(i+1) = (\cup_{k \geq n} \{bp^k(i)\}) \setminus \{0\} \cup \{i+1\}$ 。证明如下：

令  $c = i, d = i+1$ , 上述结论可以改写为：要么  $bp(i+1) = i$ , 要么  $bp(i+1) \leq bp(i)$ 。进一步的, 令  $c = bp(i), d = i+1$ , 注意到  $bp(i) = c < bp(i+1) < i$  是不可能的, 因此要么  $bp(i+1) = bp(i)$ , 要么  $bp(i+1) \leq bp^2(i)$ 。这一过程可以一直持续下去, 由数学归纳, 可知  $bp(i+1) \subseteq BP(i)$ 。

上述推论让我们对  $BP(r)$  的构成有了更直观的认识。显然, 假若对某一  $r_0$ ,  $l \notin BP(r_0)$ , 那么：1. 任意  $r > r_0$ ,  $l \notin BP(r)$ ; 2. 总存在一个最小的  $r_0 = r_m$ , 使得当  $r < r_m$  时,  $l \in BP(r)$ , 且  $BP(r_m) = \{r_m\}$ 。

## 固定 $r$

显然，当  $l$  变为  $l+1$  时，所有  $r \geq r_m$  的  $g(l, r)$  的值是不变的，接下来考虑  $r < r_m$  的  $g(l, r)$  的变化情况，为此，我们考察  $BP(r)$  的变化情况，这里，我们假定  $r_m$  足够大（否则是平凡的）。记  $BP'(r)$  为  $g(l+1, r)$  对应的选择方案的集合。

显然有  $BP'(l+1) = \{l+1\}$ ，而  $BP(l+1) = \{l, l+1\}$ ，因此  $g(l+1, l+1) = g(l, l+1) + b_l - \sum_{q_i=l} a_i$ ，令  $\Delta_l = b_l - \sum_{q_i=l} a_i$ ，可以改写为  $g(l+1, l+1) = g(l, l+1) + \Delta_l$ ，这里  $\Delta_l$  是一个“基本量”——显而易见的，任意满足  $l+1 \in BP(r)$  的  $r$ ，总存在一种可能性，满足  $BP'(r) = BP(r) \setminus \{l\}$ ，从而  $g(l+1, r) = g(l, r) + \Delta_l$ ，即不等式  $g(l+1, r) \leq g(l, r) + \Delta_l$  总是满足的。更一般的，我们可以定义  $\Delta_x = cnt(0, x) - b_x - g(l, x)$ ，这一重要变量将在后文反复出现。



## 固定 $r$

注意到  $BP(i+1) \setminus \{i+1\} \subseteq BP(i)$ , 因此存在一个最小的  $r = r_1$ , 满足当  $r < r_1$  时  $l+1 \in BP(r)$ ;  $r \geq r_1$  时  $l+1 \notin BP(r)$ 。对于  $r_1$ , 我们知道  $BP(r_1) = \{l, r_1\}$ , 显然  $BP'(r_1) = \{r_1\}$ , 因此  $g(l+1, r_1) = cnt(0, r_1) - b_{r_1} = g(l, r_1) + \Delta_{r_1}$ 。接下来考虑  $r < r_1$  的部分。这里, 根据上述讨论, 有  $g(l+1, r) \leq g(l, r) + \Delta_l$ , 我们接下来证明一个引理:

**引理 1:** 存在一个最小的  $r = r_2$ , 满足当  $r < r_2$  时  $l+1 \in BP'(r)$  即  $g(l+1, r) = g(l, r) + \Delta_l$ ;  $r \geq r_2$  时  $l+1 \notin BP'(r)$  即  $g(l+1, r) < g(l, r) + \Delta_l$ 。

## 固定 $r$

证明：设  $r_2' = \inf\{r \mid l+1 \notin BP'(r)\}$  表示最小的满足  $l+1 \notin BP'(r)$  的  $r$ ，设  $BP(r_2') = \{s_1, s_2, s_3, \dots, s_{k-1}, s_k\}$ ，其中  $s_1 = l, s_2 = l+1, s_k = r_2'$ 。注意到  $BP'(r) \setminus BP(r) = \emptyset$ （这是因为，假若  $BP'(r) \setminus BP(r) \neq \emptyset$ ，那么选择  $BP'(r) \cup \{l\}$  和  $BP'(r)$  中的较优的方案将会得到一个更优的  $g(l, r)$ ，这与最优化条件矛盾），同时  $BP'(r_2) \subseteq BP'(s_{k-1}) = BP(r_2) \setminus \{l, r_2\}$ ，因此存在一个  $n < k$ ，满足  $BP'(r_2) = \{s_2, s_3, \dots, s_{n-1}, s_n, s_k\}$ 。注意到假若  $n \geq 2$ ，那么选择  $BP'(r_2) \cup \{l\}$  同样得到一个更优的  $g(l, r)$ ，因此  $n = 1$ ，即  $BP'(r_2) = \{r_2\}$ ，即  $l+1 \notin BP'(r_2)$ 。又由  $BP(i+1) \setminus \{i+1\} \subseteq BP(i)$ ， $l+1 \notin BP'(r)$  对所有  $r > r_2$  的  $r$  也成立，引理得证。

注意到此时  $BP'(r_2) = \{r_2\}$ ，从而  $g(l+1, r_2) = g(l, r_2) + \Delta_{r_2}$ 。又由  $g(l+1, r_2) < g(l, r_2) + \Delta_l$ ，得  $\Delta_{r_2} < \Delta_l$ 。进一步的，当  $r < r_2$  时， $\Delta_r \geq \Delta_l$  总是成立。因此可以通过查找第一个满足  $\Delta_r < \Delta_l$  的  $r$  的方式找出  $r_2$ 。

## 固定 $r$

我们发现，当  $r_2 < r < r_1$  时，可以规约到  $r \leq r_2$  的情况，从而有下述引理：

**引理 2:** 当  $r < r_1$  时，设序列  $p_1, p_2, \dots, p_k$  满足  $l = p_1 < p_2 < \dots < p_k$ ,  $\forall i > 2, p_{i-1} < j < p_i$ , 均有  $\Delta_{p_i} < \Delta_{p_{i-1}} < \Delta_j$ , 且  $\forall j \in (p_k, r_1), \Delta_{p_k} < \Delta_j$ , 有：令  $M = \sup\{x \mid p_x \leq r\}$ , 那么  $g(l+1, r) = g(l, r) - \Delta_M$ 。特别地，记  $p_k = r_3$ 。

## 固定 $r$

接下来我们考虑  $r \geq r_1$  的情况，我们希望将引理 2 中  $r < r_1$  的限制去掉，即证明下述引理：

**引理 3:**  $\Delta_{r_1} \leq \Delta_{r_3}$ 。

**证明：** 考虑一种选取方案  $S = \{l, r_3\}$ ，其对应的权值为  $\text{cnt}(0, r_3) - b_{r_3} + \sum_{p_i \leq l, l \leq q_i < r_3} a_i - b_l$ ，令  $\Delta_{r_3}' = b_l - \sum_{p_i \leq l, l \leq q_i < r_3} a_i$ ，那么可以改写为  $\text{cnt}(0, r_3) - b_{r_3} - \Delta_{r_3}'$ 。根据最优化条件，这一权值不小于  $g(l, r_3)$ ，即  $\text{cnt}(0, r_3) - b_{r_3} - \Delta_{r_3}' \geq g(l, r_3) = \text{cnt}(0, r_3) - b_x - \Delta_{r_3}$ ，从而  $\Delta_{r_3} \geq \Delta_{r_3}'$ 。注意到  $r_1 > r_3$ ，因此有  $\Delta_{r_1} = b_l - \sum_{p_i \leq l, l \leq q_i < r_1} a_i \leq b_l - \sum_{p_i \leq l, l \leq q_i < r_3} a_i = \Delta_{r_3}' \leq \Delta_{r_3}$ ，因此引理得证。

## 固定 $r$

利用引理 3, 我们可以将引理 2 推广为:

设序列  $p_1, p_2, \dots, p_k$  满足  $l = p_1 < p_2 < \dots < p_k$ ,  $\forall i > 2, p_{i-1} < j < p_i$ , 均有  $\Delta_{p_i} < \Delta_{p_{i-1}} < \Delta_j$ , 且  $\forall j \in (p_k, n), \Delta_{p_k} < \Delta_j$ , 有: 令  $M = \sup\{x \mid p_x \leq r\}$ , 那么  $g(l+1, r) = g(l, r) - \Delta_M$ 。

这样, 我们完整的得到了  $g(l, *)$  推  $g(l+1, *)$  的公式。注意到对于任意元素  $r$ , 其成为某一  $p_i$  就意味着  $BP'(r) = \{r\}$ , 因此在  $l$  从 1 迭代到  $n$  的过程中会且仅会出现一次, 从而暴力找  $p$  的过程最多出现  $n$  次。综上所述, 我们可以在  $O(n \log n)$  的时间内算出任意的  $g(l, r)$ 。